# CIS 500 Software Foundations (Fall 2002)

## Homework Assignment 3

Induction; Operational Semantics

**Due:** Thursday, September 26, 2002, by 4PM

**Submission instructions:**

- Turn in your solutions in hardcopy form to Christine Metz in 556 Moore.
- If it is difficult for you to come to campus on Thursday, you may submit your solutions by *fax* to 215 898-0587. You *must also* email Christine Metz (cmetz@cis.upenn.edu) to alert her to look for your fax.
- There is no electronic submission option for this homework, contrary to what was announced in class — sorry about that!

**1 Exercise** Explain the flaw in the following "proof."

> **Theorem(?!):** All horses are the same color.
>
> **Proof:** Let $P(n)$ be the predicate "in all nonempty collections of n horses, all the horses are the same color." We show that $P(n)$ holds for all $n$ by induction on $n$.
>
> **Base case**: Clearly, $P(1)$ holds.
>
> **Induction case**: Given $P(n)$, we must show $P(n+1)$.
>
> Consider an arbitrary collection of $n+1$ horses. Remove one horse temporarily. Now we have $n$ horses and hence, by the induction hypothesis, the $n$ horses are all the same color. Now call the exiled horse back and send a different horse away. Again, we have a collection of $n$ horses and hence, by the induction hypothesis, the $n$ horses are the same color. Moreover, these $n$ horses are obviously the same color as the first collection. Thus, the horse we brought back was the same color as the horse we sent away.
>
> Therefore, all the $n+1$ horses are the same color.

**2 Exercise [Optional]** Make up your own "false proof", in which an *incorrect use of the induction hypothesis* leads to some interesting and surprising conclusion.

**3 Exercise** 3.5.13 in TAPL.

**4 Exercise** 3.5.17 in TAPL.

**5 Exercise** 4.2.2 in TAPL.

To complete this exercise, you will need to download your own copy of the files for the `arith` implementation. These files can be found on the TAPL web site

`http://www.cis.upenn.edu/∼bcpierce/tapl/checkers/arith.tar.gz` (as a single bundle)
`http://www.cis.upenn.edu/∼bcpierce/tapl/checkers/arith` (as separate files)

Instructions for compiling these files into a running implementation can be found here:

`http://www.cis.upenn.edu/∼bcpierce/tapl/resources.html#checkers`

The only changes you will need to make for this problem are to the `eval` function in the file `core.ml`. Please include a printout of *just this file* in the solutions you hand in.

**6 Debriefing**

1. How many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did you work on it mostly alone, or mostly with other people?
4. How deeply do you feel you understand the material it covers (0%–100%)?
5. Any other comments?

## Solutions

**1.** The problem with this proof is that the inductive step does not work for all $n$, namely for $n = 2$. According to our proof, to show that $n + 1$ horses are the same color, we consider horse #1 with color $C_1$ and then we separately consider horse #2 with color $C_2$. The two groups of horses have no overlapping members, and we have no reason to assert that these two horses have the same color. The argument works fine when $n > 2$, but the $n = 2$ case invalidates the induction.

**5.** The definitions of `NoRuleApplies`, `isnumericval`, and `isval` are the same as in the original; `eval1` is deleted; `eval` is rewritten as follows:

```
let rec eval t = match t with
    v when isval v -> v
  | TmIf(_,t1,t2,t3) ->
      begin
        match eval t1 with
          TmTrue _ -> eval t2
        | TmFalse _ -> eval t3
        | _ -> raise NoRuleApplies
      end
  | TmSucc(fi,t1) ->
      begin
        match eval t1 with
          nv1 when isnumericval nv1 -> TmSucc (dummyinfo, nv1)
        | _ -> raise NoRuleApplies
      end
  | TmPred(fi,t1) ->
      begin
        match eval t1 with
          TmZero _ -> t1
        | TmSucc(_, nv1) -> nv1
        | _ -> raise NoRuleApplies
      end
  | TmIsZero(fi,t1) ->
      begin
        match eval t1 with
          TmZero _ -> TmTrue(dummyinfo)
        | TmSucc(_, _) -> TmFalse(dummyinfo)
        | _ -> raise NoRuleApplies
      end
  | _ ->
      raise NoRuleApplies
```

The rest of the solutions can be found in the back of the book.